Package ‘modQR’

January 16, 2019

Encoding UTF-8

Version 0.1.2

Date 2019-01-15

Title Multiple-Output Directional Quantile Regression

Author Miroslav Šiman [aut],
Pavel Boček [aut, cre]

Maintainer Pavel Boček <bocek@utia.cas.cz>

Description Contains basic tools for performing multiple-output quantile regression and computing regression quantile contours by means of directional regression quantiles. In the location case, one can thus obtain halfspace depth contours in two to six dimensions. Hallin, M., Paindaveine, D. and Siman, M. (2010) Multivariate quantiles and multiple-output regression quantiles: from L1 optimization to halfspace depth. Annals of Statistics 38, 635-669 For more references about the method, see Help pages.

Classification/MSC 62H05, 62J99, 62G08, 65C60

Repository CRAN

Depends R (>= 2.5.0), lpSolve (>= 5.6.1), geometry (>= 0.3-1)

Suggests rgl

License LGPL-2

Collate addItem.R findItem.R delItem.R addRow.R findRow.R checkArray.R checkCTechSTu.R getCTechSTM1u.R getCTechSTM2u.R getCharSTM1u.R getCharSTM2u.R findOptimalBasisM1FromScratch.R findOptimalBasisM2FromScratch.R compContourM1u.R compContourM2u.R evalContour.R

NeedsCompilation no

Date/Publication 2019-01-16 14:50:07 UTC

R topics documented:

`compContourM1/2u` *Directional Regression Quantile Computation*

Description

The functions `compContourM1u` and `compContourM2u` may be used to obtain not only directional regression quantiles for *all* directions, but also some related overall statistics. Their output may also be used for the evaluation of the corresponding regression quantile regions by means of `evalContour`. The functions use different methods and algorithms, namely `compContourM1u` is based on [01] and [06] and `compContourM2u` results from [03] and [07]. The corresponding regression quantile regions are nevertheless virtually the same. See all the references below for further details and possible applications.

Usage

```
compContourM1u(Tau = 0.2, YMat = NULL, XMat = NULL, CTechST = NULL)
compContourM2u(Tau = 0.2, YMat = NULL, XMat = NULL, CTechST = NULL)
```

Arguments

`Tau`	the quantile level in (0, 0.5).
`YMat`	the N x M response matrix with two to six columns, `N > M+P-1`. Each row corresponds to one observation.
`XMat`	the N x P design matrix including the (first) intercept column. The default NULL value corresponds to the unit vector of the right length. Each row corresponds to one observation.
`CTechST`	the (optional) list with some parameters influencing the computation and its output. Its default value can be generated by method-dependent `getCTechSTM1/2u` and then modified by the user before its use in `compContourM1/2u`.

Details

Generally, the performance of the functions deteriorates with increasing Tau, N, M, and P as for their reliability and time requirements. Nevertheless, they should work fine at least for two-dimensional problems up to N = 10000 and P = 10, for three-dimensional problems up to N = 500 and P = 5, and for four-dimensional problems up to N = 150 and P = 3.

Furthemore, common problems related to the computation can fortunately be prevented or overcome easily.

Bad data - the computation may fail if the processed data points are in a bad configuration (i.e., if they are not in general position or if they would lead to a quantile hyperplane with at least one zero coefficient), which mostly happens when discrete-valued/rounded/repeated observations, dummy variables or bad random number generators are employed. Such problems can often be prevented if one perturbs the data with a random noise of a reasonably small magnitude before the computation, splits the model into separate or independent submodels, cleverly uses affine equivariance, or replaces a few identical observations with a copy of them weighted by the total number of their occurrences.

Bad Tau - the computation may fail for a finite number of problematic quantile levels, e.g., if Tau is an integer multiple of 1/N in the location case with unit weights (when the sample quantiles are not uniquely defined). Such a situation may occur easily for Tau's with only a few decimal digits or in a fractional form, especially when the number of observations changes automatically during the computation. The problem can be fixed easily by perturbing Tau with a sufficiently small number in the right direction, which should not affect the resulting regression quantile contours although it may slightly change the other output. The strategy is also adopted by `compContourM1/2u`, but only in the location case and with a warning output message explaining it.

Bad scale - the computation may fail easily for badly scaled data. That is to say that the functionality has been heavily tested only for the observations coming from a centered unit hypercube. Nevertheless, you can always change the units of measurements or employ full affine equivariance to avoid all the troubles. Similar problems may also arise when properly scaled data are used with highly non-uniform weights, which frequently happens in local(ly) polynomial regression. Then the weights can be rescaled in a suitable way and the observations with virtually zero weights can be excluded from the computation.

Bad expectations - the computation and its output need not meet false expectations. Every user should be aware of the facts that the computation may take a long time or fail even for moderately sized three-dimensional data sets, that the `HypMat` component is not always present in the list `COutST$CharST` by default, and that the sample regression quantile contours can be not only empty, but also unbounded and crossing one another in the general regression case.

Bad interpretation - the output results may be easily interpreted misleadingly or erroneously. That is to say that the quantile level Tau is not linked to the probability content of the sample (regression) Tau-quantile region in any straightforward way. Furthermore, any meaningful parametric quantile regression model should include as regressors not only the variables influencing the trend, but also all those affecting the dispersion of the multivariate responses. Even then the cuts of the resulting regression quantile contours parallel to the response space cannot be safely interpreted as conditional multivariate quantiles except for some very special cases. Nevertheless, such a conclusion could somehow be warranted in case of nonparametric multiple-output quantile regression; see [09].

Value

Both compContourM1u and compContourM2u may display some auxiliary information regarding the computation on the screen (if `CTechST$ReportI = 1`) or store their in-depth output (determined by `CTechST$BriefOutputI`) in the output files (if `CTechST$OutSaveI = 1`) with the filenames beginning with the string contained in `CTechST$OutFilePrefS`, followed by the file number padded with zeros to form six digits and by the extension '`.dqo`', respectively. The first output file produced by `compContourM1u` would thus be named '`DQOutputM1_000001.dqo`'.

Both compContourM1u and compContourM2u always return a list with the same components. Their interpretation is also the same (except for CharST that itself contains some components that are method-specific):

`CharST` the list with some default or user-defined output. The default one is provided by function `getCharSTM1u` for `compContourM1u` and by function `getCharSTM2u` for `compContourM2u`. A user-defined function generating its own output can be employed instead by changing `CTechST$getCharST`.

`CTechSTMsgS` the (possibly empty) string that informs about the problems with input `CTechST`.

`ProbSizeMsgS` the (possibly empty) string that warns if the input problem is very large.

`TauMsgS` the (possibly empty) string that announces an internal perturbation of `Tau`.

`CompErrMsgS` the (possibly empty) string that decribes the error interrupting the computation.

`NDQFiles` the counter of (possible) output files, i.e., as if `CTechST$OutSaveI = 1`.

`NumB` the counter of (not necessarily distinct) optimal bases considered.

`PosVec` the vector of length N that desribes the position of individual (regression) observations with respect to the exact (regression) Tau-quantile contour. The identification is reliable only after a successful computation. `PosVec[i] = 0/1/2` if the `i`-th observation is in/on/out of the contour. If `compContourM2u` is used with `CTechST$SkipRedI = 1`, then `PosVec` correctly detects only all the outer observations.

`MaxLWidth` the maximum width of one layer of the internal algorithm.

`NIniNone` the number of trials when the initial solution could not be found at all.

`NIniBad` the number of trials when the found initial solution did not have the right number of clearly nonzero coordinates.

`NSkipCone` the number of skipped cones (where an interior point could not be found).

If `CTechST.CubRegWiseI = 1`, then the last four components are calculated over all the individual orthants.

References

[01] Hallin, M., Paindaveine, D. and Šiman, M. (2010) Multivariate quantiles and multiple-output regression quantiles: from L1 optimization to halfspace depth. *Annals of Statistics* **38**, 635–669.

[02] Hallin, M., Paindaveine, D. and Šiman, M. (2010) Rejoinder (to [01]). *Annals of Statistics* **38**, 694–703.

[03] Paindaveine, D. and Šiman, M. (2011) On directional multiple-output quantile regression. *Journal of Multivariate Analysis* **102**, 193–212.

[04] Šiman, M. (2011) On exact computation of some statistics based on projection pursuit in a general regression context. *Communications in Statistics - Simulation and Computation* **40**, 948–956.

[05] McKeague, I. W., López-Pintado, S., Hallin, M. and Šiman, M. (2011) Analyzing growth trajectories. *Journal of Developmental Origins of Health and Disease* **2**, 322–329.

[06] Paindaveine, D. and Šiman, M. (2012) Computing multiple-output regression quantile regions. *Computational Statistics & Data Analysis* **56**, 840–853.

[07] Paindaveine, D. and Šiman, M. (2012) Computing multiple-output regression quantile regions from projection quantiles. *Computational Statistics* **27**, 29–49.

[08] Šiman, M. (2014) Precision index in the multivariate context. *Communications in Statistics - Theory and Methods* **43**, 377–387.

[09] Hallin, M., Lu, Z., Paindaveine, D. and Šiman, M. (2015) Local bilinear multiple-output quantile/depth regression. *Bernoulli* **21**, 1435–1466.

Examples

```
##computing all directional 0.15-quantiles of 199 random points
##uniformly distributed in the unit square centered at zero
##- preparing the input
Tau  <- 0.15
XMat <- matrix(1, 199, 1)
YMat <- matrix(runif(2*199, -0.5, 0.5), 199, 2)
##- Method 1:
COutST <- compContourM1u(Tau, YMat, XMat)
##- Method 2:
COutST <- compContourM2u(Tau, YMat, XMat)
```

`evalContour` *Evaluating Convex Polytopes*

Description

Given the system of inequalities `AAMat%*%ZVec <= BBVec` describing a convex polytope/contour with an interior point `IPVec` in the Euclidean space of dimension two to six, this function identifies all nonredundant constraints and computes some characteristics of the resulting convex polytope such as its vertices, facets, volume and surface area.

Usage

```
evalContour(AAMat, BBVec = NULL, IPVec = NULL)
```

Arguments

`AAMat`	the constraints matrix from the system of inequalities defining the convex polytope. It should be a numeric matrix with two to six columns.
`BBVec`	the right-hand side from the system of inequalities defining the convex polytope. It should be a numeric column vector of the same length as the first column of `AAMat`.
`IPVec`	an interior point of the investigated convex polytope. This argument can be omitted or set equal to a numeric column vector of the same length as the first row of `AAMat`. If `IPVec` is `NULL` or if given `IPVec` does not lie well inside the convex contour, then a well-positioned interior point is searched for internally, which may slow down the computation and make it less reliable.

Details

This function is included to be used for evaluating (regression) quantile contours or their cuts.

In fact, the function analyzes not the polytope itself, but its regularized intersection with the zero-centered hypercube of the edge length 2 000 that is employed as an artificial bounding box to avoid the problems with unbounded contours. The regularization consists of rounding the vertices (i.e., all of their coordinates) of such an intersection to the seventh decimal digit and of considering only the polytope determined by all the distinct rounded vertices for the final analysis.

Value

evalContour returns a list with the following components describing the resulting convex polytope:

`Status`
: 0 - OK.
: 2 - the contour seems virtually empty.
: 3 - the search for a well-positioned interior point `IPVec` failed.
: 4 - the number of input parameters is too low.
: 5 - `AAMat` is not a numeric matrix with two to six columns.
: 6 - `BBVec` is not a numeric column vector of the right length.
: 7 - `IPVec` is not a numeric column vector of the right length.

`TVVMat`
: the matrix with clearly distinct contour vertices (in rows).

`TKKMat`
: the matrix with clearly distinct elementary facets (in rows). Each row contains the indices of the rows of `TVVMat` where the facet vertices are stored. Each facet has the same number of vertices equal to the number of columns of `AAMat`. See also `help(convhulln)` for the meaning of `TKKMat`.

`NumF`
: the number of clearly distinct contour facets.

`NumV`
: the number of clearly distinct contour vertices.

`Vol`
: the volume of the contour (the area in 2D).

`Area`
: the surface area of the contour (the circumference in 2D and the surface in 3D).

Examples

```
##- a simple example using a tilted zero-centered square
AAMat <- rbind(c(-1,-1), c(1,-1), c(1,1), c(-1,1))
BBVec <- c(1, 1, 1, 1)
IPVec <- c(0, 0)
CST <- evalContour(AAMat, BBVec, IPVec)
print(CST)

##- computing and evaluating the 0.15-quantile contour of 199
##random points uniformly distributed in the unit square
##centered at zero
Tau  <- 0.15
YMat <- matrix(runif(2*199, -0.5, 0.5), 199, 2)
C <- compContourM1u(Tau, YMat)
CST <- evalContour(-C$CharST$HypMat[,1:2], -C$CharST$HypMat[,3])
print(CST)
```

```
##See also the examples ExampleA to ExampleE for some
##more elaborate ways of computing, evaluating and
##plotting the (regression) quantile contours.
```

getCharSTM1u *Computing Some Overall Characteristics in* compContourM1u

Description

The function computes some overall characteristics of directional regression quantiles in the output of compContourM1u, namely the list COutST$CharST. It makes possible to obtain some useful information without saving any file on the disk, and it can be easily modified by the users according to their wishes.

Usage

```
getCharSTM1u(Tau, N, M, P, BriefDQMat, CharST, IsFirst)
```

Arguments

Tau	the quantile level in (0, 0.5).
N	the number of observations.
M	the dimension of responses.
P	the dimension of regressors including the intercept.
BriefDQMat	the method-specific matrix containing the rows of a potential individual output file corresponding to CTechST$BriefOutputI = 1. See the details below.
CharST	the output list, updated with each run of the function.
IsFirst	the indicator equal to one in the first run of getCharSTM1u (when CharST is initialized) and equal to zero otherwise.

Details

This function is called inside compContourM1u. First, it is called with BriefDQMat = NULL, CharST = NULL and IsFirst = 1 to initialize the output list CharST, and then it is called with IsFirst = 0 successively for the content of each potential output file corresponding to CTechST$BriefOutputI = 1, i.e., even if the output file(s) are not stored on the disk owing to CTechST$OutSaveI = 0.

It still remains to describe in detail the content of possible output files, describing the optimal conic segmentation of the directional space that lies behind the optimization problem involved.

If CTechST$BriefOutputI = 1, then the rows of such files are vectors of length 1+1+M+M+P+1 of the form c(ConeID, Nu, UVec, BDVec, ADVec, LambdaD) where

ConeID is the number/order of the cone related to the line. If M > 2, then a cone can appear in the output repeatedly (under different numbers).

Nu is the number of corresponding negative residuals.

UVec is a normalized vector of the cone. It is usually its vertex direction but it may also be its interior vector pointing to a vertex of the artificial intersection of the cone with the bounding box `[-1,1]^M`. The max normalization is used if the breadth-first search algorithm is employed and the L2 normalization is used in the other case (when `M` = 2 and `CTechST$D2SpecI` = 1).

BDVec is the vector `c(b_1,...,b_M)`, i.e., the constant vector denominator of `BVec`, where `BVec` = `BDVec/(t(BDVec)%*%UVec)`.

ADVec is the vector `c(a_1,...,a_P)`, i.e., the constant vector denominator of `AVec`, where `AVec` = `ADVec/(t(BDVec)%*%UVec)`.

LambdaD is the constant scalar denominator of `Lambda` = `LambdaD/(t(BDVec)%*%UVec)`.

Recall that `c(BVec, AVec)` stands for the coefficients of the regression quantile hyperplane associated with `UVec` and that `Lambda` denotes the Lagrange multiplier equal to the optimal value `Psi` of the objective function for that direction.

If `CTechST$BriefOutputI` = 0, then the rows of the potential output file(s) are longer (of length `1+1+M+M+P+1+(P+M-1)*M+(P+M-1)`) because they contain two more vectors appended at the end. The rows are of the form `c(ConeID, Nu, UVec, BDVec, ADVec, LambdaD, vec(VUMat), IZ)` where

VUMat is the matrix for computing the multiplier vector `MuR0Vec` associated with zero residuals, `MuR0Vec = (VUMat%*%UVec)/(t(BDVec)%*%UVec)`. That is to say that all directions from the interior of the cone result in the regression `Tau`-quantile hyperplanes containing the same `P+M-1` observations because all such hyperplanes are the same up to a scaling factor multiplying their coefficients.

IZ is the vector containing original indices of the `M+P-1` observations with zero residuals for all directions from the interior of the cone.

Value

`getCharSTM1u` returns a list with the following components:

`NUESkip`	the number of (skipped) directions (and corresponding hyperplanes) artificially induced by intersecting the cones with the `[-1,1]^M` bounding box.
`NAZSkip`	the number of (skipped) hyperplanes (and corresponding directions) not counted in NUESkip and with at least one coordinate of `AVec` zero.
`NBZSkip`	the number of (skipped) hyperplanes (and corresponding directions) not counted in NUESkip and with at least one coordinate of `BVec` zero.
`HypMat`	(for `M > 4`) the component is missing (for `M <= 4`) the matrix with `M + P` columns containing (in rows) all the distinct regression `Tau`-quantile hyperplane coefficients `c(BVec, AVec)` normalized with `|BVec|`, rounded to the eighth decimal digit, and sorted lexicographically. This matrix can be used for the computation of the regression `Tau`-quantile contour.
`CharMaxMat`	the matrix with the (slightly rounded) maxima of certain directional regression `Tau`-quantile characteristics over all remaining vertex directions. If `P` = 1, then `CharMaxMat` has only three rows: `c(UVec, max(|BVec|)),`

`c(UVec, max(Lambda))`, and

`c(UVec, max(Lambda/|BVec|))`,

respectively.
If `P` > 1, then the rows of `CharMaxMat` are as follows:

`c(UVec, max(|BVec|))`,

`c(UVec, max(Lambda))`,

`c(UVec, max(Lambda/|BVec|))`,

`c(UVec, max(|c(a_2,...,a_P)|))`,

`c(UVec, max(|c(a_2,...,a_P)|/|BVec|))`,

`c(UVec, max(|a_2|))`,

`c(UVec, max(|a_2|/|BVec|))`,

`...`,

`c(UVec, max(|a_P|))`, and

`c(UVec, max(|a_P|/|BVec|))`,

respectively. If `P` = 2, then the last two rows are missing for not being included twice.

`CharMinMat` the matrix with the (slightly rounded) minima of certain directional regression Tau-quantile characteristics over all remaining vertex directions.
If `P` = 1, then `CharMinMat` has only three rows:

`c(UVec, min(|BVec|))`,

`c(UVec, min(Lambda))`, and

`c(UVec, min(Lambda/|BVec|))`,

respectively.
If `P` > 1, then `CharMinMat` has five rows:

`c(UVec, min(|BVec|))`,

`c(UVec, min(Lambda))`,

`c(UVec, min(Lambda/|BVec|))`,

`c(UVec, min(|c(a_2,...,a_P)|))`, and

`c(UVec, min(|c(a_2,...,a_P)|/|BVec|))`,

respectively.

Note that `||` symbolizes the Euclidean norm, and that the vertices (`UVec`) in the rows of `CharMaxMat` and `CharMinMat` are generally different and denote (one of) the direction(s) where the row maximum or minimum is attained.

Examples

```
##Run print(getCharSTM1u) to examine the default setting.
```

getCharSTM2u *Computing Some Overall Characteristics in* `compContourM2u`

Description

The function computes some overall characteristics of directional regression quantiles in the output of `compContourM2u`, namely the list `COutST$CharST`. It makes possible to obtain some useful information without saving any file on the disk, and it can be easily modified by the users according to their wishes.

Usage

```
getCharSTM2u(Tau, N, M, P, BriefDQMat, CharST, IsFirst)
```

Arguments

`Tau`	the quantile level in (0, 0.5).
`N`	the number of observations.
`M`	the dimension of responses.
`P`	the dimension of regressors including the intercept.
`BriefDQMat`	the method-specific matrix containing the rows of a potential individual output file corresponding to `CTechST$BriefOutputI` = 1. See the details below.
`CharST`	the output list, updated with each run of the function.
`IsFirst`	the indicator equal to one in the first run of `getCharSTM2u` (when `CharST` is initialized) and equal to zero otherwise.

Details

This function is called inside `compContourM2u`. First, it is called with `BriefDQMat` = `NULL`, `CharST` = `NULL` and `IsFirst` = 1 to initialize the output list `CharST`, and then it is called with `IsFirst` = 0 successively for the content of each potential output file corresponding to `CTechST$BriefOutputI` = 1, i.e., even if the output file(s) are not stored on the disk owing to `CTechST$OutSaveI` = 0.

It still remains to describe in detail the content of possible output files, describing the optimal conic segmentation of the directional space that lies behind the optimization problem involved.

If `CTechST$BriefOutputI` = 1, then the rows of such files are vectors of length `1+1+M+P*M+M` of the form `c(ConeID, Nu, UVec, vec(ACOMat), MuBRow)` where

ConeID is the number/order of the cone related to the line. If `M` > 2, then a cone can appear in the output repeatedly (under different numbers).

Nu is the number of negative residuals corresponding to the interior directions of the cone.

UVec is a normalized vector of the cone. It is usually its vertex direction but it may also be its interior vector pointing to a vertex of the artificial intersection of the cone with the bounding box `[-1,1]^M`. The max normalization is used if the breadth-first search algorithm is employed and the L2 normalization is used in the other case (when `M` = 2 and `CTechST$D2SpecI` = 1).

ACOMat is the matrix describing AVec, AVec = ACOMat%*%UVec.

MuBRow is the constant vector of the Lagrange multipliers corresponding to BVec. Its inner product with UVec is equal to the optimal value Psi of the objective function for that direction.

Recall that c(BVec, AVec) stands for the coefficients of the regression quantile hyperplane associated with UVec and always BVec = UVec.

If CTechST$BriefOutputI = 0, then the rows of the potential output file(s) are longer (of length 1+1+P*M+M+P+P) because they contain two more vectors appended at the end. The rows are of the form c(ConeID, Nu, UVec, vec(ACOMat), MuBRow, MuR0Row, IZ) where

MuRORow is the constant vector of the Lagrange multipliers corresponding to zero residuals associated with the interior of the cone. That is to say that all directions from the interior of the cone result in the regression Tau-quantile hyperplanes containing the same P observations.

IZ is the vector containing original indices of the P observations with zero residuals for all directions from the interior of the cone.

Value

getCharSTM2u returns a list with the following components:

NUESkip — the number of (skipped) directions (and corresponding hyperplanes) artificially induced by intersecting the cones with the [-1,1]^M bounding box

NAZSkip — the number of (skipped) hyperplanes (and corresponding directions) not counted in NUESkip and with at least one coordinate of AVec zero.

NBZSkip — the number of (skipped) hyperplanes (and corresponding directions) not counted in NUESkip and with at least one coordinate of BVec zero.

HypMat — (for M > 4) the component is missing
(for M <= 4) the matrix with M + P columns containing (in rows) all the distinct regression Tau-quantile hyperplane coefficients c(BVec, AVec) rounded to the eighth decimal digit and sorted lexicographically. This matrix can be used for the computation of the regression Tau-quantile contour.

CharMaxMat — the matrix with the (slightly rounded) maxima of certain directional regression Tau-quantile characteristics over all remaining vertex directions.
If P = 1, then CharMaxMat has only two rows:

c(UVec, max(Psi)), and
c(UVec, max(|MuBRow|)),

respectively.
If P > 1, then the rows of CharMaxMat are as follows:

c(UVec, max(|Psi|)),
c(UVec, max(MuBRow)),
c(UVec, max(|c(a_2,...,a_P)|)),
c(UVec, max(|a_2|)),
...,
c(UVec, max(|a_P|)),

respectively. If P = 2, then the last row is missing for not being included twice.

`CharMinMat` the matrix with the (slightly rounded) minima of certain directional regression Tau-quantile characteristics over all remaining vertex directions.
If `P` = 1, then `CharMinMat` has only two rows:

`c(UVec, min(Psi))`, and

`c(UVec, min(|MuBRow|))`,

respectively.
If `P` > 1, then `CharMinMat` has three rows:

`c(UVec, min(Psi))`,

`c(UVec, min(|MuBRow|))`, and

`c(UVec, min(|c(a_2,...,a_P)|))`,

respectively.

Note that `||` symbolizes the Euclidean norm, and that the vertices (`UVec`) in the rows of `CharMaxMat` and `CharMinMat` are generally different and denote (one of) the direction(s) where the row maximum or minimum is attained.

Examples

```
##Run print(getCharSTM2u) to examine the default setting.
```

`getCTechSTM1/2u` *Getting the List of Options CTechST for* `compContourM1/2u`

Description

The functions `getCTechSTM1u` and `getCTechSTM2u` set the default list of options CTechST for computing all the directional (regression) quantiles by means of `compContourM1u` and `compContourM2u`, respectively.

Usage

```
getCTechSTM1u()
getCTechSTM2u()
```

Arguments

none

Details

Fortunately, the default list of options usually leads to a satisfactory performance in all but very large problems.

Value

Both `getCTechSTM1u` and `getCTechSTM2u` produce a list with a few components whose default values are stated below after the equality sign.

The components `OutFilePrefS` and `getCharST` are initialized in a method-specific way.

The components `CubRegWiseI`, `ArchAllFI`, and `SkipRedI` are relevant only if `D2SpecI` is zero or if the dimension of directions/responses is higher than two, i.e., if the breadth-first search algorithm is used.

Most of the components are generated by both functions. Nevertheless, the component `SkipRedI` is only generated by `getCTechSTM2u` and used by `compContourM2u`.

The output components are as follows:

`ReportI` = 0; if some information (such as the progress of computation) is displayed on the screen (1) or not (0). The display mode may slightly slow down the computation, especially when the dimension of responses is higher than two. On the other hand, it shows the new value of the quantile level (Tau) (if the input one has been changed internally), the initial L2-normed directional vector used (`U0Vec`), the number of failures to find an initial solution (`NNotFound`), the number of found initial solutions not having the right number of clearly nonzero coordinates (`NBad`), and also the width of each layer of the breadth-first search algorithm if it is employed.

`OutSaveI` = 0; if the detailed output is stored in file(s) into the working directory (1) or not (0). The file output seems necessary only for very large problems if some information about individual cones has to be recorded (such as all the regression quantile hyperplanes used for the regression quantile contour computation).

`D2SpecI` = 1; this option is relevant only for bivariate directions/responses and determines if the cones are visited counter-clockwise (1) or by means of the breadth-first search algorithm as in the general case (0). The default option (1) leads to a more precise and reliable computation than the other.

`BriefOutputI` = 1; if the brief (1) or verbose (0) output is prepared by `compContourM1/2u`. Even the default option (1) is sufficient for almost all common applications. See also `getCharSTM1u` and `getCharSTM2u` for the description of the possible method-specific file output in both cases.

`CubRegWiseI` = 1; if the directional space is divided into orthants investigated separately (1) or not (0). On the one hand, the default option (1) splits the problem into smaller ones. On the other hand, it also generates some artificial cones with at least one facet in the orthant borders.

`ArchAllFI` = 1; if all the past cone facet identifiers (1) or only those from the last few layers (0) are stored during the computation. The default option (1) makes the computation more likely to terminate successfully than the other. Unfortunately, it is also slower and more memory demanding. If the dimension of responses is higher than three, then `ArchAllFI` = 1 is considered internally by `compContourM1/2u` no matter what the input `CTechST` actually says.

`SkipRedI` = 0; if the information should be skipped (1) or stored (0) also from the cones with all non-artificial facets already known (such cones are redundant/irrelevant with probability one if only all the quantile regression hyperplanes necessary for the quantile contour computation are required from compContourM2u). The skipping makes the output smaller but maybe also slightly less reliable. It also affects the reliability of the information regarding the inner points; see compContourM2u.

OutFilePrefS = 'DQOutputM1_'/'DQOutputM2_'; the prefix of possible output file name(s).

getCharST = getCharSTM1u/getCharSTM2u; the function computing some overall characteristics that can be replaced with a user-defined one. See getCharSTM1u and getCharSTM2u for the default choices.

Examples

```
##- a typical use of getCTechSTM1u:
##computing all directional 0.01-quantiles of 49 random points
##(uniformly distributed in the unit cube centered at zero)
##after changing the default settings
Tau  <- 0.01
XMat <- matrix(1, 49, 1)
YMat <- matrix(runif(3*49, -0.5, 0.5), 49, 3)
CTechST <- getCTechSTM1u()
CTechST$ReportI <- 1
COutST <- compContourM1u(Tau, YMat, XMat, CTechST)
```

Index